JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 23, No. 12, pp. 2527-2536, Dec. 2022

Check for updates

서로 다른 색상 공간을 가지는 스테레오 이미지에서 마스크를 착용한 사람들에 대한 3차원 공간 정보 획득

박 태 정*
*덕성여자대학교 사이버보안전공 부교수

Obtaining 3D Spatial Information about People Wearing Masks from Stereo Images with Different Color Spaces

Taejung Park*

*Associate Professor, Department of Cyber Security/Information Technology and Media Engineering, College of Engineering, Duksung Women's University, Seoul 01369, Korea

[요 약]

본 논문에서는 별도의 신뢰할 수 있는 깊이 정보 없이 동시에 RGB 이미지와 적외선(IR) 이미지 획득이 가능한 스테레오 카메라 시스템을 이용해서 마스크를 착용한 상태로 화면에 주어진 시각 정보를 응시하는 사람들의 모습을 촬영한 대규모 데이터세트에서 피실험자들의 눈과 얼굴 각 지점에 대한 3차원 정보를 추출하는 방법을 논의한다. 본 논문에서 다루는 카메라 시스템은 두 카메라의 색상 공간이 다르기 때문에 널리 사용되는 이미지 프로세싱 알고리즘들을 바로 적용할 수 없으며 마스크 착용으로 인한 여러 가지 한계도 발생한다. 따라서 본 연구에서는 얼굴의 랜드마크(landmark)을 포착할 수 있는 알고리즘을 적용하여 두 이미지 간의 일치점을 파악하였다. 또한 랜드마크에 대해 3차원 지점을 근사하기 위한 알고리즘도 제안한다. 약 250,000 쌍에 달하는 데이터들을 처리하기 위해서 CPU 기반 병렬처리를 수행하였으며 그 결과 인공지능 기반 시선 추적 알고리즘을 학습시키기에 충분한 수준인 93.95%에 달하는 데이터에서 3차원 기하 정보를 추출할 수 있었다.

[Abstract]

This paper presents a solution to calculate positions in three-dimensional space for a stereo camera system that can acquire RGB and infrared (IR) images at the same time without reliable depth information, for a large-scale image of people gazing at the specific positions on the screen while wearing a mask. In the camera system dealt with in this paper, since the color spaces of the two cameras are different, widely used image processing algorithms cannot be directly applied, and various limitations occur due to wearing a mask. To address this, an algorithm that can capture a specific point of the face was applied to identify the coincidence point between the two images. This paper also proposes an algorithm for approximating three-dimensional points. CPU-based parallel processing has been performed to process about 250,000 pairs of data to get three-dimensional geometric information with success rate of 93.95%, which is sufficient to train an AI-based eye tracking algorithm.

색인어 : 시선추적, 키넥트, 스테레오 카메라, 인공지능 데이터세트, AI-Hub
Keyword : Gaze Estimation, Kinect, Stereo Camera, Artificial Intelligence Datasets, AI-Hub

http://dx.doi.org/10.9728/dcs.2022.23.12.2527

Received 09 November 2022; **Revised** 23 November 2022
Accepted 14 December 2022

***Corresponding Author; Taejung Park**

Tel: +82-2-901-8339
E-mail: tjpark@duksung.ac.kr

Ⅰ. 서 론

인공지능 기술의 발전과 함께 그동안 독립적으로 발전해 왔던 Microsoft Kinect, Asus Xtion, Intel RealSense 등의 깊이 정보 기반 하드웨어 센서의 활용 영역이 크게 확대되고 있다. 이러한 깊이 정보 기반 하드웨어 센서는 전형적으로 일반적인 RGB 카메라와 함께 적외선(IR) 카메라와 ToF(Time of Flight) 기술을 결합하여 픽셀 단위의 깊이 정보를 제공한다는 측면에서 RGBD (Red Green Blue Depth) 카메라라고 통칭하기도 한다.

이러한 유형의 센서는 기본적으로 두 개의 카메라(RGB와 IR)를 장착하고 있기 때문에 본질적으로 스테레오 카메라 시스템으로 볼 수 있다. 동시에 픽셀 단위의 정밀한 깊이 정보를 활용할 수 있기 때문에 3차원 기하 정보 스캔, 인체 동작 감지, 가상 카메라 시점 변환 등 다양한 응용 분야를 지원한다. 그러나 IR 카메라를 기반으로 한 깊이 측정 방식은 태양광이나 할로겐 조명 등과 같이 적외선 잡음이 많은 환경에서 올바로 작동하지 않으며[1] 깊이 정보의 오변환으로 인해 정보가 소실되는 경우도 발생한다.

본 논문에서는 위와 같은 이유로 인해 깊이 정보 기반 하드웨어 센서로 측정된 대규모 데이터에서 깊이 정보를 신뢰할 수 없거나 사용할 수 없는 경우 RGB 이미지와 IR 이미지 사이의 공통 지점을 파악한 후 스테레오 카메라 시스템을 활용하여 epipolar geometry를 기초로 이미지 상의 2차원 공통 지점 좌표들의 3차원 공간 좌푯값을 계산하는 방법을 논의한다. 또한 실제로 AI Hub[2]에서 제공하는 공개 데이터들 중 시선 추적을 위해 Azure Kinect로 피실험자의 얼굴을 중심으로 촬영한 데이터세트에 논의된 방법을 적용한 실험 결과를 제시한다.

1-1 관련 연구

1) 시선 추적

시선 추적(gaze estimation)은 사용자의 얼굴을 촬영한 2차원 이미지만으로 해당 사용자가 바라 보고 있는 위치를 예측하는 기술이다. 이러한 시선 추적 기술은 크게 모델 기반(model-based) 방법과 외관 기반(appearance-based) 방법으로 나눌 수 있다[3].

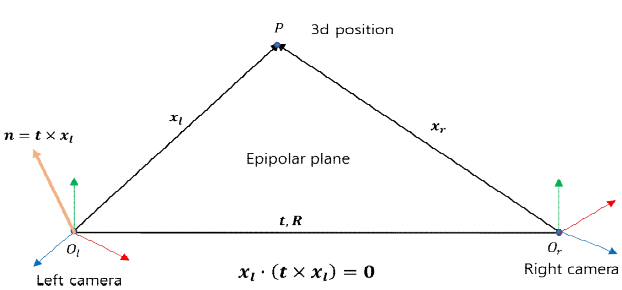

그림 1. Epipolar 기하학
Fig. 1. Epipolar geometry

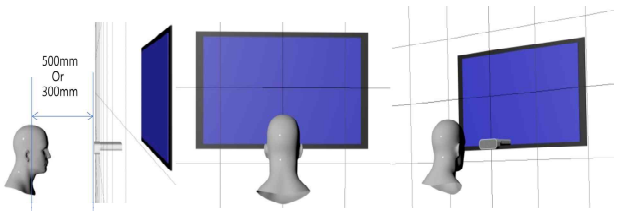

그림 2. AI HUB 데이터세트 데이터 캡처 설정
Fig. 2. Configuration for capturing the AI HUB dataset

모델 기반 시선 추적 또는 예측 기법은 주로 눈에 가까운 위치에 근적외선(NIR) 광원과 카메라를 배치해서 안구 표면에 반사된 형상을 이용하는 방법이다. 이 방법은 주로 눈의 해부학적인 형태를 기하학적인 모델로 재구성해서 눈이 바라보는 곳을 추정한다. 모델 기반 시선 추적 기법은 직관적인 특징이 있지만 근적외선 광원을 안구에 가깝게 위치시킴으로써 화상을 유발하는 문제와 함께 고해상도 적외선 카메라 등 고가의 전용 장비가 필요하다는 단점이 있다.

이에 비해 외관 기반 시선 추적 기법은 인공지능 기법을 기반으로 일반적인 웹캠 등 상대적으로 해상도가 낮은 카메라를 이용해서 사용자의 얼굴을 촬영하고 인공지능을 이용해서 시선 방향을 추적하는 기술이다. 전용 장비가 필요하지 않기 때문에 응용 범위가 넓다는 장점이 있으나 딥러닝 네트워크를 학습시키기 위해서 대규모 학습 데이터가 필요하다는 단점이 있다.

본 연구에서는 외관 기반 시선 추적 기법의 학습을 위해 서로 다른 색 공간과 카메라 매개변수를 가지는 두 이미지에서 3차원 정보를 추출하여 학습 데이터를 구성하는 방법에 대해 논의한다.

2) Epipolar Geometry

Epipolar geometry는 일정한 거리를 두고 두 대의 카메라로 동시에 3차원 공간 상의 한 물체를 촬영하는 경우의 기하해석 방법이라고 할 수 있다.

그림 1에서 왼쪽 카메라의 위치 O_l와 오른쪽 카메라 위치 O_r를 연결하는 벡터 $\boldsymbol{t}$, 3차원 공간 상의 한 점 P를 연결하는 삼각형이 포함되는 평면을 epipolar plane이라고 한다. 이 때 벡터 $\boldsymbol{t}$와 왼쪽 카메라의 위치 O_l에서 P로 향하는 벡터 $\boldsymbol{x_l}$과의 외적 $\boldsymbol{n}=\boldsymbol{t}\times\boldsymbol{x_l}$는 수직이기 때문에 $\boldsymbol{x_l}\cdot(\boldsymbol{t}\times\boldsymbol{x_l})=0$가 성립하는데 이 방정식을 epipolar equation이라고도 한다. 이 방정식을 이용하면 스테레오 카메라 시스템에서 점 P의 3차원 공간 상에서의 위치를 구할 수 있다. 자세한 내용은 [4]를 참고한다.

Ⅱ. 본 론

2-1 데이터세트

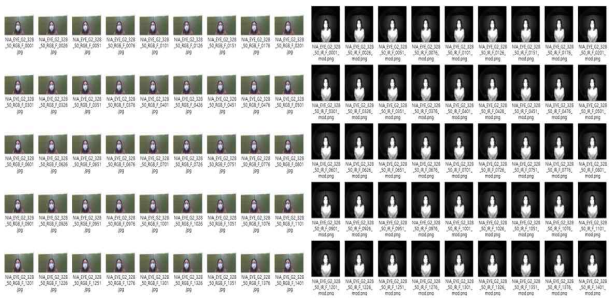

그림 3. RGB 이미지와 IR 이미지로 구성된 데이터 샘플
Fig. 3. Data samples that have both RGB and IR images

본 논문에서는 AI HUB에서 공개한 안구 움직임 영상 데이터[2]를 다룬다. 이 영상 데이터 중 일부에서는 피실험자의 머리의 중심을 IR 카메라로부터 300mm와 500mm 거리로 위치한 상태에서 피실험자가 화면에 무작위로 지시된 한 점을 응시하는 모습을 Microsoft Azure Kinect를 사용해서 RGB, Depth, IR(적외선) 영상으로 동시에 촬영한 데이터가 포함된다(그림 2). 특히 Depth 정보의 경우 Azure Kinect의 카메라 모드에 따라 2mm에서부터 5460mm까지의 범위로 mm 단위로 측정해서 16비트 unsigned int(uint16)로 저장되어야 한다. uint16 형식으로 저장될 경우 카메라의 실제 측정 가능 거리와 상관 없이 최대 0mm~65,535mm (약 65m)까지 거리 정보를 mm 단위로 저장할 수 있다. 그러나 논문 작성 중인 2022년 10월을 기준으로 실제로 AI HUB에 업로드된 Depth 정보는 8비트 unsigned int (int8)로 저장되어 정확한 depth 정보의 획득이 어려운 상황이다.

Azure Kinect는 RGB 카메라와 IR 카메라 사이의 여러 가지 연산(예. inter calibration, virtual camera projection) 뿐만 아니라 RGB 카메라와 3차원 공간 정보의 계산(예. point cloud, 3차원 기하 정보 대응 등)을 위해서 depth 정보를 활용하기 때문에 depth 정보가 부정확하다면 이 데이터세트의 유용성이 크게 감소할 수 밖에 없다.

특히 원래 이 데이터세트가 의도하고 있는 3차원 시선 정보의 계산이라는 근원적인 목적도 달성하기 어렵다. 정확한 depth 정보가 제공된다면 하드웨어의 지원으로 소프트웨어 연산 부담을 줄이면서 피실험자의 머리 방향(head rotation)과 눈의 3차원 위치, 시선 벡터를 실시간으로 소프트웨어 연산으로 인한 오차를 줄이면서 비교적 정확하게 추정할 수 있지만 depth 정보가 없는 이러한 상황에서는 RGB 카메라와 IR 카메라를 스테레오 카메라 시스템으로 해석해서 3차원 정보를 유추할 수 밖에 없는 한계가 있다. 본 논문에서는 이러한 문제를 극복하고자 그림 3에서 제시한 것과 같은 RGB 이미지와 IR 이미지만을 이용해서 3차원 정보를 구성하는 방법을 논의한다.

2-2 스테레오 카메라 시스템에서의 두 이미지 간 일치점 파악

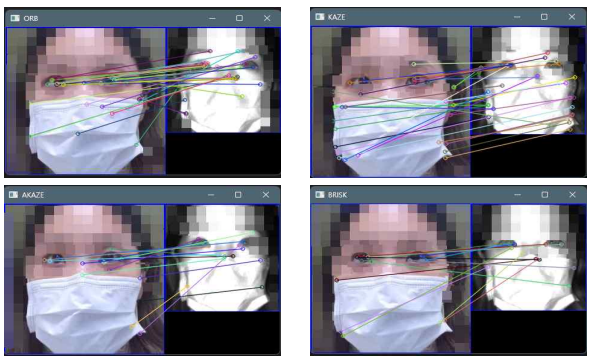

그림 4. RGB 이미지와 IR 이미지 사이의 일치점 검색 결과(ORB, KAZE, AKAZE, BRISK)
Fig. 4. Matching results between RGB and IR images (ORB, KAZE, AKAZE, BRISK)

스테레오 카메라 시스템은 두 카메라가 고정된 위치와 각도로 배치되어 동시에 같은 장면을 촬영하는 시스템을 의미한다. 이렇게 촬영된 이미지들은 이후에 논의할 epipolar coordinates로 해석되어 이미지에서 공통되는 2차원 지점들에 대한 3차원 공간 정보를 계산할 수 있다.

두 이미지 사이에서 2차원 일치점(matching point)을 찾을 수 있는 알고리즘으로는 SIFT, SURF, ORB, KAZE, AKAZE, BRISK 등이 있다. 이 알고리즘들 중에서 특허로 인해서 OpenCV 라이브러리로 사용할 수 없는 SIFT, SURF를 제외한 나머지 알고리즘을 이용해서 RGB 이미지와 IR 이미지 사이의 일치점들을 계산하였다(그림 4).

결과를 살펴보면 동일한 카메라로 구성되는 일반적인 스테레오 카메라 시스템과는 달리 Azure Kinect의 경우 RGB 이미지와 IR 이미지 사이의 일치점을 찾아야 하는 한계로 인해서 모든 알고리즘에서 일치점들이 정확하게 파악되지 않았다. RGB 이미지의 경우 빨간색, 초록색, 파란색 등 세 개의 채널로 픽셀별로 상대적으로 풍부한 정보를 이용할 수 있지만 IR 이미지의 경우 IR 센서의 감응 정도(전압)을 각 픽셀의 음영으로 표현하는 한계로 인해 일반적으로 RGB-RGB 이미지 사이에서의 일치점을 찾기 위해 개발된 알고리즘들은 신뢰할 수 없는 결과를 제공하는 것으로 판단된다. 따라서 두 이미지 사이의 정확한 일치점을 찾을 수 없기 때문에 일반적인 일치점 찾기 알고리즘은 본 논문에서 의도하는 목적으로 사용할 수 없다.

2-3 MediaPipe를 이용한 일치점 파악

Google MediaPipe[5]는 얼굴 감지, 얼굴 매시 생성, 홍채 인식, 손가락/신체 자세 감지, 머리카락 영역 분할(segmentation), 물체 감지, 움직임 추적 등 여러 작업들을 수행할 수 있는 인공지능 기반 기술이다. 특히 인간의 얼굴에 적용하는 경우 BlazeFace[6] 기술을 기반으로 모든 사람들에게 공통적인 랜드마크(landmark)를 찾고 이 지점들을 기준으로 마스크 형태의 얼굴 매시(face mesh)를 생성하는 기능을 수행한다(그림 5).

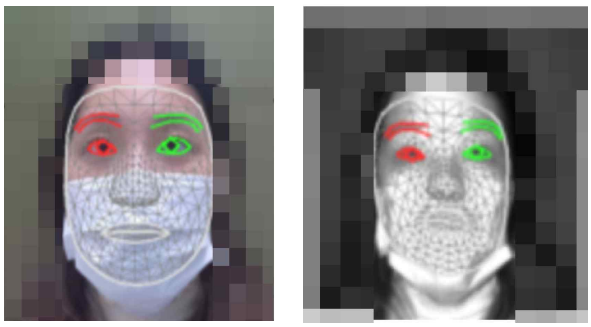

그림 5. RGB 이미지(왼쪽)와 IR 이미지(오른쪽)에 MediaPipe를 적용한 결과

Fig. 5. Results of applying MediaPipe to RGB image (left) and IR image (right)

앞서 살펴 본 대로 IR 이미지와 RGB 이미지 사이의 일반적인 일치점 파악 알고리즘으로는 올바른 결과를 얻기 힘든 것(그림 4)에 비해 MediaPipe를 IR 이미지와 RGB 이미지에 적용한 결과, 두 이미지에서 모두 비교적 안정적으로 특정 지점을 얻을 수 있었다(그림 5). 특히 AI HUB의 안구 움직임 영상 데이터의 경우 모든 피실험자들이 마스크를 착용하고 있어서 랜드마크가 전혀 파악되지 않은 경우도 있었고 약간의 오차는 불가피한 측면이 있지만, MediaPipe가 본 연구에서 중요한 눈 주변에서의 공통적인 특징적 지점을 비교적 정확하게 찾기 때문에 피실험자의 3차원 위치 정보(특히 눈 부분)를 계산하기 위해 이후에 설명할 3차원 위치 추정 알고리즘에 적용할 수 있었다.

2-4 카메라 매개변수와 렌즈 왜곡 보정

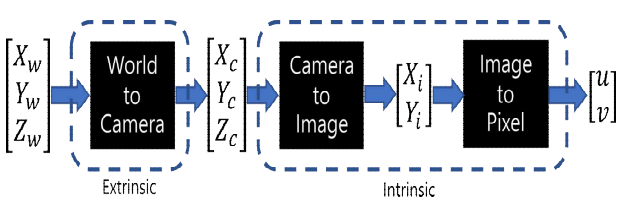

그림 6. 카메라 좌표 변환

Fig. 6. Camera coordinates transformation

Azure Kinect는 IR 카메라와 RGB 카메라가 고정된 위치와 각도로 배치되는 형태를 가지고 있으며 이 위치와 각도에 의해서 외부 카메라 매개변수(extrinsic camera parameter)가 결정된다(그림 6). 또한 내부 카메라 매개변수(intrinsic camera parameter)를 통해서 각 렌즈의 왜곡률, 초점 등의 정보를 확인할 수 있다[7]. Azure Kinect 공식 사이트[8]에 외부 카메라 매개변수를 결정하는 두 카메라의 상대적인 위치와 각도가 명시되어 있으나 정밀도가 떨어지며 Azure Kinect SDK[9]를 통해 기기를 연결한 후 정확한 매개변수를 읽어 올 수 있다.

1) Extrinsic Camera Parameters

Azure Kinect에서 외부 카메라 매개변수는 IR 카메라의 중점을 기준으로 RGB 카메라가 얼마나 떨어져 있으며(translation), 어떤 각도(rotation)로 바라보고 있는지 정보를 제공한다. Azure Kinect는 Y축의 양의 방향이 지면을 향하고 Z축의 양의 방향이 카메라가 보는 방향(view direction)을 향하는 오른손 좌표계를 채택하고 있으며 IR 카메라가 원점에 위치하며 RGB 카메라는 Azure Kinect 사양에서 약 [32.0, 1.8, -6.34] (단위 mm) 지점에 위치하면서 RGB 카메라가 IR 카메라보다 X축 기준으로 약 +6도 회전한 형태로 배치되어 있다(이 수치는 대략적인 수치이며 정확한 수치는 기기가 연결된 상태에서 Azure Kinect SDK를 통해 얻을 수 있다).

2) Intrinsic Camera Parameters

그림 6에서 볼 수 있듯이 내부 카메라 매개변수는 3차원 카메라 좌표계에서 2차원 이미지 좌표계로 변환(주로 원근투사)하고 카메라 왜곡을 반영하여 2차원 이미지 좌표계를 2차원 픽셀 좌표계로 변환하기 위해 사용된다.

일반적으로 카메라 좌표계에서의 3차원 좌표 (X_c, Y_c, Z_c)는 초점 거리가 1인 바늘구멍 카메라 모델에서 다음과 같이 2차원 이미지 좌표 (X_i, Y_i)로 변환된다[10].

$$X_i = \frac{X_c}{Z_c},\ Y_i = \frac{Y_c}{Z_c} \tag{1}$$

물리적인 카메라에는 이상적인 카메라와는 달리 제조 상의 한계로 인한 왜곡이 존재할 수 밖에 없는데 이러한 렌즈 왜곡을 표현하고 해석하기 위해서 왜곡 함수 $d(r)$로 왜곡된 이미지 좌표 (X_d, Y_d)를 다음과 같이 모델링한다.

$$X_d = X_i d(r),\ Y_d = Y_i d(r) \tag{2}$$

$$r = \sqrt{X_i^2 + Y_i^2}$$

Azure Kinect에서는 OpenCV와 마찬가지로 r에 대한 6차 유리함수를 이용하는 Brown Conrady 왜곡 모델[11]을 사용한다. 최종적인 픽셀 좌표 (u,v)는 다음과 같이 계산한다.

$$u = s_x X_d + X_0, v = s_y Y_d + Y_0 \tag{3}$$

이때 s_x, s_y는 픽셀/초점 거리(focal distance) 단위로 표현된 배율로 이미지의 해상도가 $w \times h$이면 $w/h = s_y/s_x$을 만족한다[10]. 또한 (X_0, Y_0)는 이상적인 바늘구멍 카메라의 이미지 좌표 (u,v)에서 이미지의 실질적 중점(principal point)의 좌표를 의미한다.

3) 렌즈 왜곡 보정

식 (2)와 (3)은 이상적인 카메라에서의 2차원 좌표 (X_i, Y_i)를 왜곡된 좌표 (X_d, Y_d)로 변환하는 과정을 보여준다. 따라서 왜곡 과정(distort)은 r에 대한 6차원 유리함수로 표현되는 왜곡 함수 d(r)를 바로 계산할 수 있지만 왜곡을 제거하는 과정(undistort)는 6차원 유리함수로 표현되는 방정식을 풀어야 하는 비선형적인 해법을 거쳐야 한다. Azure Kinect SDK에서는 이 과정을 최대경사하강법(steepest gradient descent)와 유사한 반복 해결법(iterative solver)를 이용해서 해결한다(Algorithm 1의 (A)).

2-4 스테레오 이미지를 이용한 3차원 위치 계산

1) 최소 z 거리와 최대 z 거리 사이 직선의 교점을 이용한 3차원 위치 복원

카메라에 투사된 3차원 좌푯값을 정확하게 알기 위해서는 식 (1)에서처럼 카메라 좌표계에서 3차원 좌표의 z축값인 Z_c를 정확하게 알아야 한다. 일반적인 Azure Kinect 기기에서는 이 값을 깊이값(depth) 정보로 정확하게 측정할 수 있지만 AI HUB의 데이터세트처럼 depth 정보를 신뢰할 수 없거나 아예 없는 경우는 바로 Z_c를 알 수 없다.

따라서 본 논문에서는 깊이값에 해당되는 정확한 Z_c를 추정하기 위해서 각 카메라에 대해 해당 3차원 지점이 가장 가깝게 위치할 때의 깊이 50mm부터 가장 멀리 위치할 때의 거리 14,000mm의 연속 범위에 대해 고정된 (X_i, Y_i)에 대한 3차원 위치를 고려한다. 이때 한 카메라에 대해 고정된 (X_i, Y_i)를 기준으로 식 (1)에 대해 $50 \le Z_c \le 14000$ 범위로 연속으로 깊이값을 변화시키면 3차원 공간에서 (X_c, Y_c, Z_c)의 점의 자취는 각 카메라에서 가장 가까운 점 $(50X_i, 50Y_i, 50)$과 가장 먼 점 $(14000X_i, 14000Y_i, 14000)$을 연결하는 선분이 된다(Algorithm 1의 (B)). 구하고자 하는 실제 3차원 좌표 (X_c, Y_c, Z_c)는 두 카메라에서 3차원 공간 상에 각각 형성된 두 선분의 교점이 된다는 것을 알 수 있다.

2) 3차원 공간에서 만나지 않는 두 직선 사이의 최소 거리 구하기

앞서 이 논문에서 제안한 3차원 위치 복원 방식으로 계산한 두 선분은 최소 깊이에서 계산된 3차원 지점과 최대 깊이에서 계산된 3차원 지점을 지나는 선분으로 계산이 된다. 그러나 실제로 32비트 또는 64비트 실수 연산으로 인한 한계와 카메라 왜곡 모델에서의 오차 등으로 두 선분이 수치적으로 계산했을 때 정확하게 한 점에서 교차하는 경우는 매우 드물다. 따라서 실용적인 근사로서 3차원 공간 상에서 만나지 않는 두 선분의 최단 거리를 통과하는 제3의 선분을 구하고 그 선분의 중점이나 (IR 카메라의 중심이 world coordinates의 기준점임을 고려해서) IR 카메라의 선분과 제3의 선분이 교차하는 점을 두 선분이 만나는 3차원 지점으로 근사한다.

이후 논의에서는 계산의 편의상 두 선분을 IR 카메라와 RGB 카메라에서 각각 가장 가까운 점 P_1과 P_2에서 출발해서 가장 먼 점을 향하는 ray(반직선)로 가정한다.

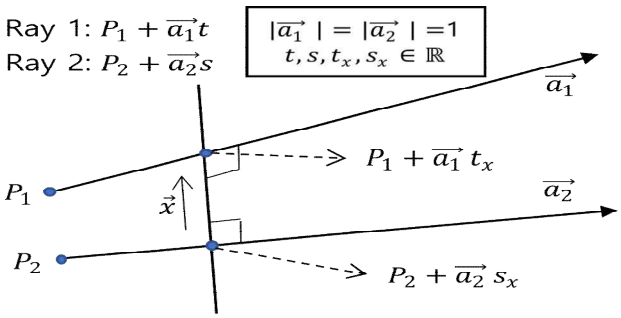

그림 7. 두 ray에서 가장 가까운 지점을 통과하는 선분
Fig. 7. The segment that passes two closest points between two rays

그림 7에서처럼 만나지 않는 두 ray가 각각 점 P1과 P2에서 출발해서 $\overrightarrow{a_1}, \overrightarrow{a_2}$ 방향으로 진행하는 ray라고 한다면 이 두 ray는 다음 식으로 표현할 수 있다.

$$ray1 : P_1 + \overrightarrow{a_1}t$$
$$ray2 : P_2 + \overrightarrow{a_2}s$$
$$\left|\overrightarrow{a_1}\right| = \left|\overrightarrow{a_2}\right| = 1, \quad t, s \in \mathbb{R}$$

이 때 최단 지점을 통과하는 선분이 지나는 지점을 각각 $P_1 + \overrightarrow{a_1}t_0, P_2 + \overrightarrow{a_2}s_0$라고 하면 이 두 지점을 지나는 직선 l은 ray1 및 ray2와 직교해야 한다. 이 때 직선 l의 방향 벡터를 $\vec{x}$라고 하면

$$u\vec{x} = \overrightarrow{a_1} \times \overrightarrow{a_2}, \quad u \in R$$
$$\vec{x} = (P_2 + \overrightarrow{a_2}s) - (P_1 + \overrightarrow{a_1}t)$$

외적의 성질에 의해

$$\overrightarrow{a_1} \bullet (\overrightarrow{a_1} \times \overrightarrow{a_2}) = 0 \tag{4}$$

$$\overrightarrow{a_2} \bullet (\overrightarrow{a_1} \times \overrightarrow{a_2}) = 0 \tag{5}$$

(4)에서

$$\overrightarrow{a_1} \bullet (\overrightarrow{a_1} \times \overrightarrow{a_2}) = \overrightarrow{a_1} \bullet u\vec{x}$$
$$= u\overrightarrow{a_1} \bullet (P_2 - P_1 + s\overrightarrow{a_2} - t\overrightarrow{a_1})$$
$$= u\left\{\overrightarrow{a_1} \bullet (P_2 - P_1) + s\overrightarrow{a_1} \bullet \overrightarrow{a_2} - t)\right\} = 0, \forall u \in \mathbb{R}$$

$$\therefore \overrightarrow{a_1} \bullet (P_2 - P_1) + s\overrightarrow{a_1} \bullet \overrightarrow{a_2} - t = 0$$

같은 방법으로 (5)에서

$$\therefore \overrightarrow{a_2} \cdot (P_2 - P_1) - t\overrightarrow{a_1} \cdot \overrightarrow{a_2} + s = 0$$

두 벡터 $\overrightarrow{a_1}, \overrightarrow{a_2}$가 이루는 각을 θ라고 하고 두 방정식을 행렬식으로 표현하면

$$Ax = \begin{bmatrix} 1 & -\overrightarrow{a_1} \cdot \overrightarrow{a_2} \\ \overrightarrow{a_1} \cdot \overrightarrow{a_2} & -1 \end{bmatrix} \begin{bmatrix} t_x \\ s_x \end{bmatrix} = \begin{bmatrix} \overrightarrow{a_1} \cdot (P_2 - P_1) \\ \overrightarrow{a_2} \cdot (P_2 - P_1) \end{bmatrix} \quad (6)$$

$$\cos\theta = \overrightarrow{a_1} \cdot \overrightarrow{a_2}$$

미지수 t_x와 s_x의 값을 알기 위해 양변에 역행렬을 곱하면

$$\therefore \begin{bmatrix} t_x \\ s_x \end{bmatrix} = \frac{1}{\cos^2\theta - 1} \begin{bmatrix} -1 & \cos\theta \\ -\cos\theta & 1 \end{bmatrix} \begin{bmatrix} \overrightarrow{a_1} \cdot (P_2 - P_1) \\ \overrightarrow{a_2} \cdot (P_2 - P_1) \end{bmatrix} \quad (7)$$

이때 정방행렬 A의 판별식은 $\cos^2\theta - 1$이며 이 값이 0이 되는 θ는 $0 \le \theta < 2\pi$ 범위에서 0, π이다. 그런데 IR 카메라의 중심에서 시작해서 카메라 좌표계의 z 축 방향으로 임의의 거리에 있는 이미지의 각 픽셀의 샘플링 지점(일반적으로 사각형 픽셀 영역의 중점)을 통과하는 ray들 중에서 임의의 두 ray는 같은 방향으로 또는 반대 방향으로 평행할 수 없기 때문에 θ는 0, π가 될 수 없다. 따라서 정방행렬 A는 역행렬을 가지며 식 (7)와 같이 미지수 t_x, s_x를 구할 수 있다.

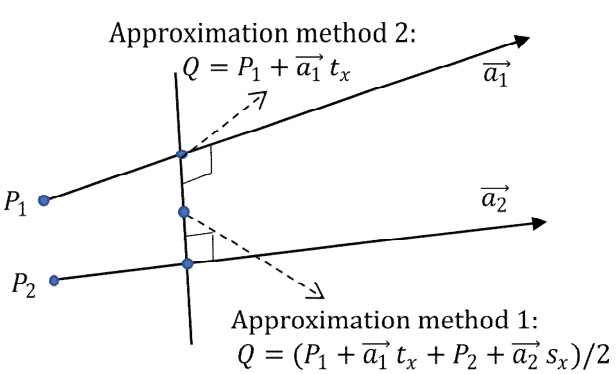

그림 8. 공간 상의 3차원 지점을 근사하는 두 가지 방법
Fig. 8. Two methods to approximate a three-dimensional position

이러한 계산 결과를 토대로 두 가지 교차점의 근사 방식을 생각해 볼 수 있다(그림 8). 첫 번째 방법은 식 (7)를 통해 구한 t_x, s_x로 ray1과 ray2 상에서 가장 가까운 두 지점을 지나는 선분을 구하고 이 중점을 교차점의 근사값으로 계산하는 방법이다. 식 (7)를 통해 구한 대로 ray1 위에서 ray2와 가장 가까운 점의 좌표는 $P_1 + \overrightarrow{a_1} t_x$이며 ray2 위에서 ray1가 가장 가까운 점의 좌표는 $P_2 + \overrightarrow{a_2} s_x$이므로 이 두 점의 중점을 두 ray의 교차점의 좌표로 가정한다면 교차점의 좌표 Q는 다음과 같다.

$$Q = (P_1 + \overrightarrow{a_1} t_x + P_2 + \overrightarrow{a_2} s_x)/2 \quad (8)$$

이 방법은 IR 카메라와 RGB 카메라에서 나온 ray가 모두 픽셀 샘플링 위치를 통과할 때 두 카메라 모두 어느 정도 오차를 가지고 있다고 가정하고 있으며 그러한 오차를 가정해서 평균으로 교차점을 결정한다고 생각할 수도 있다.

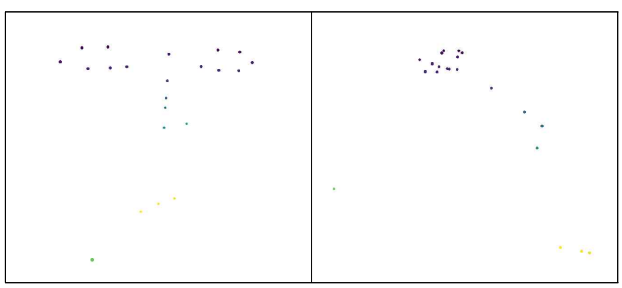

그림 9. 그림 5에서 제시된 이미지에서의 랜드마크에 대한 3차원 위치 정보 재구성 결과. 왼쪽 – 정면, 오른쪽 – 측면
Fig. 9. 3D positions reconstructed for the landmark points in the image in Figure 5. left - front, right - side

이와는 다르게 IR 카메라의 중심을 world coordinates의 원점으로 가정하고 있다는 점에 착안해서 간단하게 IR 카메라에서 나온 ray 위의 가장 가까운 점을 교차점으로 가정하는 방법도 생각해 볼 수 있다. 이때 ray1이 IR 카메라에서 나오는 ray라고 한다면 근사 교차점 Q는 다음과 같다.

$$Q = P_1 + \overrightarrow{a_1} t_x \quad (9)$$

Ⅲ. 실험 및 결과

3-1 대규모 데이터 계산을 위한 병렬처리

표1에서 제시한 Algorithm 1은 전체 데이터세트에서 동시에 촬영된 IR 이미지와 RGB 이미지를 지금까지 논의한 방법에 적용해서 두 눈의 3차원 위치를 병렬적으로 계산하는 의사코드이다.

표1에서 제시한 Algorithm 1은 Python과 C++를 이용해서 구현하였고 계산 대상이 되는 데이터세트는 IR/RGB 이미지로 구성된 약 250,000 쌍 정도의 큰 규모이기 때문에 AMD EPYC CPU 2개(3.0GHz)가 설치된 Linux 서버에서 64개 CPU 코어(물리적 코어 32개)를 동시에 이용하는 병렬처리를 수행하였다. 방대한 데이터를 처리하는 중에 파악하기 힘든 MediaPipe나 기타 구현 상의 문제로 인한 오류로 인해 작업이 중단되는 경우를 파악하고자 예외처리(Python의 try/exception)를 이용해서 로그를 작성하였고 재시작 시 이

미 처리되었거나 오류가 발생한 데이터는 기록으로 남겨 두고 처리하지 않은 데이터를 계속해서 처리할 수 있도록 코드를 작성하였다.

병렬처리 결과, 전체 데이터에서 약 93.95%는 의도한대로 정상적으로 작동하였으나 6.05% 정도의 데이터는 다음에서 논의할 두 가지 문제로 인하여 신뢰할 수 없는 결과로 나타났다.

표 1. 대규모 IR 이미지/RGB 이미지 쌍에서 3차원 좌우 눈 좌표를 계산하기 위한 병렬처리 의사 코드

Table 1. Pseudo code for parallel processing to compute the 3D positions of the left and right eyes in a large number of IR and RGB image pairs

Algorithm 1: Parallel 3D Eye Position Extractor eye_positions(I, R)

Input:
- 2D IR Image Set I = {i_1, ...i_n} with resolution 1024x1024
- 2D RGB Image Set R= {r_1, ...r_n} with resolution 1280x720 for m subjects and p sequences/subject (n =mp)

An image pair (i_k, r_k) has been taken simultaneously with an Azure Kinect device.

Output:
- 3D Eye Positions (Left and Right) reconstructed for each image pair (i_k, r_k)

```
process find_3d_position(ir_point2d, rgb_point2d):
 // --- undistort 2d points ---- (A)
  undist_ir_2d = undistort(ir_point2d)
  undist_rgb_2d  = undistort(rgb_point2d)

 // --- find the closest and the farest 3d points ---- (B)
  min_ir_3d = (50*undist_ir_2d.x, 50*undist_ir_2d.y, 50)
  max_ir_3d = (14000*undist_ir_2d.x, 14000*undist_ir_2d.y, 14000)
  min_rgb_3d = (50*undist_rgb_2d.x, 50*undist_rgb_2d.y, 50)
  max_rgb_3d = (14000*undist_rgb_2d.x, 14000*undist_rgb_2d.y, 14000)

 // --- calculate 2 rays ---- (C)
  ray1 = create_ray(min_ir_3d, max_ir_3d)
  ray2 = create_ray(min_rgb_3d, max_rgb_3d)

 // find the closest point ---- (D)
  Q = find_closest(ray1, ray2)
  return Q
end

process find_closest(ray1, ray2):
  tx, sx =  solve Equation (7)
  closest = (P1 + a1 tx + P2 + a2 sx)/2 or P1 + a1 tx
  return closest

main process eye_positions(I, R):
  foreach_parallel(Ik, Rk):

 // get 2d eye landmarks on each image using MediaPipe---- (E)
    ir_right_eye_landmarks, ir_left_eye_landmarks = MediaPipe(Ik)
    rgb_right_eye_landmarks, rgb_left_eye_landmarks = MediaPipe(Rk)

 // get 3d eye landmark positions
 // using landmark pairs on IR and RGB images---- (F)
    foreach({left|right}_eye_2d_landmarks):
      3d_landmark_{left|right}_positions
         = find_3d_position({left|right}_ir_point2d, {left|right}_rgb_point2d)
    3d_left_eye_position = average(3d_landmark_left_positions)
    3d_right_eye_position = average(3d_landmark_right_positions)

    return 3d_left_eye_position, 3d_right_eye_position
```

3-2 3차원 정보 재구성 결과

그림 9는 그림 5에서 제시한 IR 및 RGB 이미지를 제안하는 결과로 처리하여 얻은 랜드마크에 대한 3차원 공간에서의 렌더링 결과이다. 이 결과를 보면 두 눈 주변의 랜드마크 위치는 정확하게 파악하는 것을 볼 수 있지만 코와 입, 턱선은 3차원 공간에서 정확하게 계산되지 않는 것을 볼 수 있다.

표 2. 그림 5에 제시한 이미지로 구성한 랜드마크의 3차원 위치

Table 2. Reconstructed 3D eye positions of landmark points on the images shown in Figure 5

	Position (mm)	Depth (mm)
3D position of left eye	(−30.87, −19.51, 402.22)	402.22
3D position of right eye	(29.57, −21.81, 414.38)	414.38

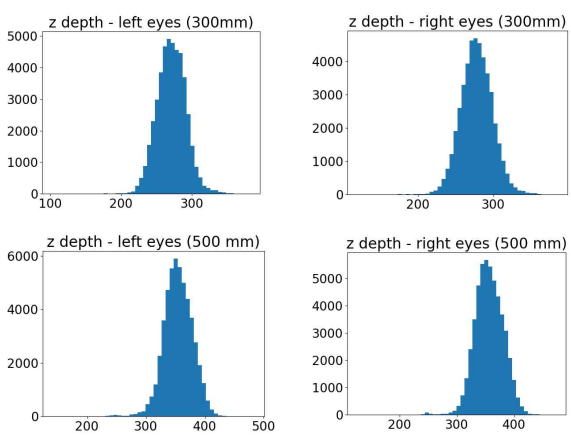

그림 10. 각 측정 거리(300mm, 500mm)와 좌우 눈에 대한 z축 방향 깊이의 분포

Fig. 10. Distribution of z-direction depth for each measurement distance (300mm, 500mm) and left and right eyes

그 이유는 그림 5에서 MediaPipe가 눈의 위치는 올바로 파악하지만 마스크로 인해서 코와 입, 턱 선의 위치는 올바로 파악하지 못하기 때문이다. 그러나 본 연구의 목적은 시선 추적 인공지능 네트워크를 학습시키기 위한 눈의 3차원 위치 정보를 파악하는 것이기 때문에 마스크로 인한 오류는 본 연구의 목적에 큰 영향을 미치지 않는다.

표 2에서는 그림 5와 그림 9를 통해 재구성한 양쪽 눈의 중심 위치의 좌표와 이 중심 위치로부터 IR 카메라의 z축 방향 깊이값을 제시한다. 표 2에서 볼 수 있는 것처럼 왼쪽 눈과 카메라 사이의 z 방향 거리(depth)는 402.22mm였고 오른쪽 눈의 경우 414.38mm였다. 그림 5의 경우 피실험자와 IR 카메라 사이의 z 축 방향 거리가 500mm라고 명시되어 있고 평균적인 인체의 머리 크기와 자세 등을 고려해 보면 합당한 거리라고 볼 수 있다.

그림 10은 각 측정 거리와 좌우 눈에 따른 z축 방향 깊이의 분포를 제시한다. 이 결과와 실제 촬영된 피실험자의 자세를 보면, 데이터 제공업체가 제공한 실험 환경(그림 2)에서는 막연하게 피실험자의 위치와 카메라 사이의 거리가 각각 300mm, 500mm 일 때로 명시하고 있으나 이 거리가 정확하게 어떤 지점인지 밝히고 있지 않으며 데이터 수집 시 다른 시선 추적 연구 데이터 수집[12]에서처럼 머리 위치를 고정시키지 않고 모니터/카메라 쪽으로 몸을 숙인다던지, 등을 기울이는 등 몸을 자유롭게 움직일 수 있는 상태로 유지하고 측정을 수행한 것으로 보인다. 따라서 개인적 차이(나이, 자세, 습관, 체형 등)에 따라 이러한 편차가 발생하는 것을 볼 수 있으나 모두 겉보기 기반 시선 추적 네트워크를 학습시키기에는 유효한 범위라고 판단된다.

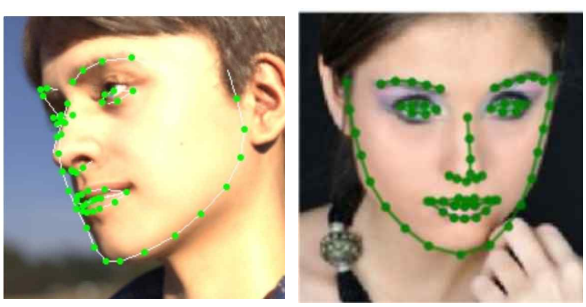

그림 11. 최신 연구 결과에서 랜드마크 오류 사례. 왼쪽 – DECA, 오른쪽 – SynergyNet

Fig. 11. Examples of landmark errors in the latest research. Left – DECA, right – SynergyNet

3-3 MediaPipe의 문제로 인한 한계

실험 결과 대부분의 이미지 쌍에 대해서는 비교적 정확한 두 눈의 3차원 좌표를 구할 수 있었으나 이 데이터세트의 모든 피실험자들이 마스크를 착용하고 촬영한 사실과 주로 RGB 이미지를 대상으로 학습된 MediaPipe를 IR 이미지에 적용함으로써 발생하는 한계로 인해 문제가 발생하였다.

얼굴 상의 랜드마크를 포착하는 경우 여러 연구 결과에서 MediaPipe와 유사하게 마스크를 착용한 경우 오류가 발생하거나 눈의 위치가 미세하게 달라지는 문제가 흔하게 발생한다. 그림 11은 Microsoft에서 다양한 얼굴 데이터 확보를 위한 합성 기법[13]으로 생성한 사람의 얼굴 위에 최신 얼굴 랜드마크 생성 기술인 DECA[14]를 적용한 모습(왼쪽)과 SynergyNet[15]의 결과(오른쪽)를 제시한다. 이 결과들을 보면 눈 주변의 랜드마크가 특징적으로 아래로 내려 가는 모습을 볼 수 있는데 현재 다른 최신 관련 기술에서도 이러한 유사한 문제가 나타나고 있다. MediaPipe 역시 이러한 한계가 있으나 다른 방법과 다르게 모바일 환경에서 실시간으로 작동 가능할 정도로 연산 부하가 적기 때문에 본 연구에서 수행한 대규모 데이터 처리에 적합하다.

본 연구에서는 대규모 데이터를 병렬로 처리하기 때문에 하나하나 세부 오류들을 파악하기 힘든 한계가 있으나 사람들의 평균적인 두 눈의 간격이 60mm~70mm 사이임을 이용해서 이 범위를 지나치게 벗어나는 결과는 오류로 판단하고 해당 사례들을 별도로 저장하였다.

1) 마스크 착용으로 인한 landmark 인식 오류

MediaPipe는 사람들이 시각적으로 불편을 느끼지 않는 조명 환경 속에서 안경이나 기타 물품을 착용하지 않은 상태로 학습되었기 때문에 얼굴을 가리는 요소가 있거나 과도한 얼굴 각도(예를 들어 카메라에 얼굴 옆 모습이 촬영되어 두 눈이 모두 보이지 않는 경우 등)로 촬영되면 오작동을 하는 경향이 있다. 특히 마스크를 착용할 경우, MediaPipe가 눈 주변의 landmark는 비교적 정확하게 파악하지만 턱이나 입 주변의 landmark에서 오류가 발생한다. 실험에 사용된 데이터세트의 경우 모든 피실험자들이 마스크를 착용한 상태로 촬영했기 때문에 일부 데이터에서는 landmark가 큰 오차로 잘못 인식되는 현상들이 발생하였다(그림 12). 본 논문에서 해결하고자 하는 문제에서는 눈 주변의 landmark만 필요하기 때문에 대부분의 데이터에서는 어느 정도 정확한 계산이 가능하였지만, 마스크 착용으로 인해 눈 부분도 어느 정도 영향이 있으리라 생각된다. 향후 마스크로 인한 랜드마크 오류 문제는 COVID-19 상황 이후 전세계적으로 여러 방법들이 제안, 실험되고 있기 때문에 향후 적절한 방식을 적용하여 해결을 시도하고자 한다.

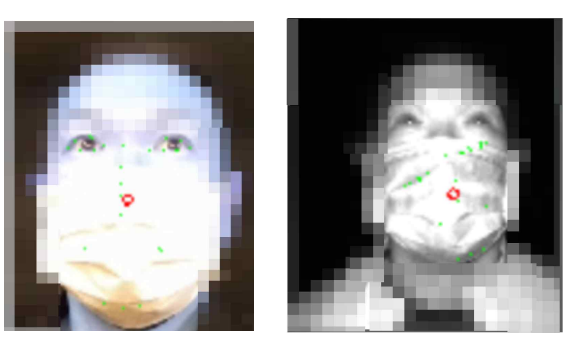

그림 12. RGB 이미지에서 정확하게 포착된 랜드마크(왼쪽)과 IR 이미지에서 잘못 포착된 랜드마크(오른쪽)

Fig. 12. Accurately captured landmarks in the RGB image (left) and incorrectly captured landmarks in the IR image (right)

2) IR 이미지에서의 landmark 인식 오류

또한 MediaPipe는 일반적인 RGB 기반의 카메라로 촬영된 얼굴 이미지에서 올바로 작동하도록 설계되었기 때문에 일부의 경우 동시에 촬영된 RGB 이미지에서는 landmark가 잘 인식되지만 IR 이미지에서는 landmark가 아예 인식되지 않은 경우도 확인할 수 있었다. 이 문제는 향후 추가적인 연구를 통하여 IR 이미지를 자연스러운 RGB 이미지로 변환하는 방식을 구현함으로써 해결하고자 한다.

표 3. 전체 데이터 대비 학습에 사용 가능한 유효한 데이터의 비율
Table 3. Ratio of valid data available for training to total data

Item	Number	Ratio
# of IR images	259,223	100%
# of RGB Images	259,223	100%
# of Success Cases	243,551	93.95%
# of Failed Cases	15,672	6.05%

Ⅳ. 결 론

깊이 정보를 별도로 사용할 수 없는 상황에서 IR 카메라와 RGB 카메라로 구성된 스테레오 카메라 시스템에서 동시에 마스크를 착용한 여러 사용자의 시선 추적을 위해 촬영된 IR 이미지와 RGB 이미지 쌍에서 3차원 시선 정보를 계산하는 방법을 제안한다.

일반적으로 동일한 종류의 카메라 두 대를 사용하는 경우와는 달리 본 실험에 사용한 Azure Kinect 기기에는 카메라 매개변수가 다르고 센서의 특성도 다른 RGB 카메라와 IR 카메라가 장착되어 있어 일반적인 스테레오 카메라에서의 3차원 정보 획득 방법을 그대로 사용할 수 없다. 특히 단순히 1개 채널에서 센서의 감응 전압 크기로 표현되는 IR 이미지의 컬러 공간은 RGB 이미지와 크게 차이가 있기 때문에 스테레오 카메라 시스템으로 동시에 촬영한 두 RGB 이미지에서 일치점을 찾는 일반적인 알고리즘을 그대로 사용할 수 없다.

이러한 한계를 극복하기 위해서 사람의 얼굴에서 2차원 랜드마크(landmark)를 찾는 알고리즘을 사용했으며 IR 이미지와 RGB 이미지에서 공통 랜드마크들에서 각각의 카메라에서 반복(iterative) 알고리즘으로 렌즈 왜곡을 배제한 이후 이론적으로 가장 가까운 거리와 가장 먼 거리를 연속적으로 연결하는 선분을 구하고 이 두 선분의 3차원 공간 상에서의 교점 혹은 가장 근접한 점을 찾음으로써 해당 2차원 랜드마크를 3차원 공간에서 구하는 수식을 제시하였다.

이렇게 눈을 구성하는 각 랜드마크의 3차원 위치를 찾은 후 두 눈에 대해 각각 랜드마크의 평균을 구해서 두 눈의 중심의 3차원 지점을 구하였다. 이후 각 이미지와 함께 기록된 사용자가 바라보는 모니터 위의 2차원 위치를 실험 공간 정보를 바탕으로 3차원 지점으로 변환한 후 눈의 3차원 위치에서 모니터 상의 표시점에 대한 3차원 위치로 향하는 벡터를 구함으로써 시선 방향(gaze direction)을 구하는 방법을 제안한다.

본 논문에서 논의한 과정은 인공지능 네트워크의 학습을 위한 대규모 시선 방향 정보를 획득하기 위한 목적이 있기 때문에 250,000 여 쌍의 이미지에 대해 시선 방향을 계산하였다. 이러한 대규모 데이터의 계산을 위해서 전체 데이터를 64개 코어로 분할하여 병렬처리를 수행함으로써 보다 짧은 시간 내에 시선 방향 데이터를 추출하였다.

병렬처리 결과 얼굴 랜드마크(landmark)을 추출하는 MediaPipe가 마스크를 착용한 경우와 IR 이미지에 대해 어느 정도 취약한 문제를 드러내는 경우를 포착하고 분리하였다. 문제 발생 여부는 병렬처리 중에 랜드마크를 아예 찾지 못하거나 랜드마크를 찾았다고 하더라도 논의한 방법에 의해 계산한 두 눈의 3차원 위치 사이의 거리가 평균을 과도하게 벗어난 경우를 검사하여 발견하였다.

문제가 발생한 경우는 전체 데이터 수에 비해서 상대적으로 미미한 수준(전체 데이터 중 6.05%, 표 3)이기 때문에 올바로 눈의 위치를 찾은 데이터만으로도 외관 기반 시선 예측(appearance-based gaze prediction)을 위한 학습 데이터로 활용이 가능하다. 본 연구를 통해 추출한 데이터는 향후 인공지능 기반 외관 기반 시선 예측 연구에 활용할 예정이다.

감사의 글

본 연구는 덕성여자대학교 2021년도 교내연구비 지원에 의해 수행되었음.

참고문헌

[1] T. Park. (2012). “Efficient Filtering for Depth Sensors under Infrared Light Emitting Sources,” *Journal of Digital Contents Society*, Vol. 13, No. 3, pp. 271-278. Sep. 2012. http://dx.doi.org/10.9728/dcs.2012.13.3.271

[2] AI Hub [Internet]. Available: https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=548

[3] E. Lindén, J. Sjöstrand, and A. Proutiere, “Learning to Personalize in Appearance-Based Gaze Tracking,” arXiv:1807.00664 [cs], Sep. 2019, Available: http://arxiv.org/abs/1807.00664

[4] K. Hata and S. Savarese, CS231A Course Note 3: Epipolar Geometry, CS231A: Computer Vision, From 3D Reconstruction to Recognition, Stanford University [Internet] Available: https://web.stanford.edu/class/cs231a/course_notes/03-epipolar-geometry.pdf

[5] Google MediaPipe [Internet]. Available: https://google.github.io/mediapipe/

[6] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, “BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs,” Jul. 2019, doi: 10.48550/arXiv.1907.05047.

[7] Aqeel Anwar, “What are Intrinsic and Extrinsic Camera Parameters in Computer Vision?”, Towards Data Science [Internet]. Available: https://towardsdatascience.com/what-are-intrinsic-and-extrinsic-camera-parameters-in-computer-vision-7071b72fb8ec

[8] Azure Kinect DK Hardware Specifications [Internet]. Available: https://learn.microsoft.com/en-us/azure/kinect-dk/hardware-specification

[9] Azure Kinect Sensor SDK [Internet]. Available: https://github.com/microsoft/Azure-Kinect-Sensor-SDK

[10] Carlo Tomasi, CPS 296.1 Supplementray Lecture Notes, Computer Vison, Duke University [Internet]. Available: https://courses.cs.duke.edu/fall07/cps296.1/CameraCalibration.pdf

[11] Brown, Duane C. "Decentering Distortion of Lenses," *Photogrammetric Engineering*. Vol. 32, No. 3, pp. 444–462. May 1966.

[12] Y. Sugano, Y. Matsushita, and Y. Sato, “Learning-by-Synthesis for Appearance-Based 3D Gaze Estimation,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, pp. 1821–1828, Jun. 2014. doi: 10.1109/CVPR.2014.235.

[13] E. Wood et al., “Fake It Till You Make It: Face analysis in the wild using synthetic data alone,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021*, pp. 3681-3691. Oct. 2021.

[14] Y. Feng, H. Feng, M. J. Black, and T. Bolkart, “Learning an Animatable Detailed 3D Face Model from in-the-Wild Images,” in *ACM Transactions on Graphics*, Vol. 40, No. 4, pp. 1–13, August 2021. https://doi.org/10.1145/3450626.3459936

[15] C.-Y. Wu, Q. Xu, and U. Neumann, “Synergy between 3DMM and 3D Landmarks for Accurate 3D Facial Geometry,” in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10442-10451, 2022. https://doi.org/10.1109/3DV53792.2021.00055

박태정(Taejung Park)

1997년 : 서울대 전기공학부 (공학사)
1999년 : 서울대 전기공학부 대학원 (공학 석사, 반도체 물리 전공)
2006년 : 서울대 전기컴퓨터공학부 대학원 (공학박사, 컴퓨터 그래픽스 전공)

2018년～현 재: 덕성여자대학교 공과대학 사이버보안/IT미디어공학과 부교수
2013년～2017년: 덕성여자대학교 정보미디어대학 디지털미디어학과 조교수
2006년～2013년: 고려대학교 연구교수
※관심분야 : 컴퓨터그래픽스, 병렬처리, 인공지능, 수치해석, 3차원 모델링